Day1: Database Design and Basic Operations

1. Database Schema Design

```
CREATE DATABASE Inventory_and_fleet;
USE Inventory_and_fleet;
```

Table Vehicale

```
CREATE TABLE Vehicles(
vehicle_id INT PRIMARY KEY,
vehicle_type VARCHAR(50),
capacity INT,
license_plate VARCHAR(20) UNIQUE,
status ENUM('available','in-service') DEFAULT 'available'
);
```

Table Drivers

```
CREATE TABLE Drivers (
driver_id INT PRIMARY KEY,
name VARCHAR(90),
license_number VARCHAR(50),
assigned_vehicle_id INT,
FOREIGN KEY (assigned_vehicle_id) REFERENCES Vehicles(vehicle_id)
);
```

Table Warehouse

```
CREATE TABLE Warehouses(
warehouse_id INT PRIMARY KEY,
location VARCHAR(100),
capacity INT
);
```

Table Inventory

```
CREATE TABLE Inventory(
item_id INT PRIMARY KEY AUTO_INCREMENT,
item_name VARCHAR(100),
stock_quantity INT,
warehouse_id INT,
FOREIGN KEY (warehouse_id) REFERENCES Warehouses(warehouse_id)
);
```

Table Deliveries

```
CREATE TABLE Deliveries(
delivery_id INT PRIMARY KEY,
vehicle_id INT,
driver_id INT,
warehouse_id INT,
delivery_date DATE,
status ENUM('pending', 'in-progress', 'completed') DEFAULT 'pending'
);
```

Table Delivery_Items

CREATE TABLE Delivery_Items(

delivery_id INT,

item_id INT,

quantity INT

);

2. Data Insertion

```
55
56 •  INSERT INTO Vehicles (vehicle_id, vehicle_type, capacity, license_plate, status) VALUES
57    (101,'Truck', 50, 'ABC123', 'available'),
58    (102,'Van', 20, 'DEF456', 'available'),
59    (103,'Motorcycle', 1, 'GHI789', 'available'),
60    (104,'Van', 15, 'MNO345', 'available'),
61    (105,'Truck', 20, 'JKL012', 'available');
62
63
64 •  INSERT INTO Drivers (driver_id, name, license_number, assigned_vehicle_id) VALUES
65    (1,'John Doe', 'LIC123456', 101),
66    (2,'Jane Smith', 'LIC789101', 102),
67    (3,'Tom Brown', 'LIC112131', 103),
68    (4,'Lucy Green', 'LIC415161', 104),
69    (5,'Emma White', 'LIC718192', 105);
70
71 •  INSERT INTO Warehouses (warehouse_id, location, capacity) VALUES
72    (3021,'Mumbai', 5000),
73    (3022,'Delhi', 4000),
74    (3023,'Hyderabad', 3500);
75
76 •  INSERT INTO Inventory (item_id,item_name, stock_quantity, warehouse_id) VALUES
77    (311,'Laptop', 50, 3022),
78    (312,'Phone', 100, 3023),
79    (313,'Tablet', 50, 3023),
80    (314,'Headphones', 100, 3021),
81    (315,'Monitor', 70, 3021),
82    (316,'Keyboard', 100, 3023),
83    (317,'Mouse', 100, 3022),
84    (318,'Charger', 100, 3022),
85    (319,'Webcam', 150, 3023),
86    (320,'Speakers', 90, 3021);
87
```

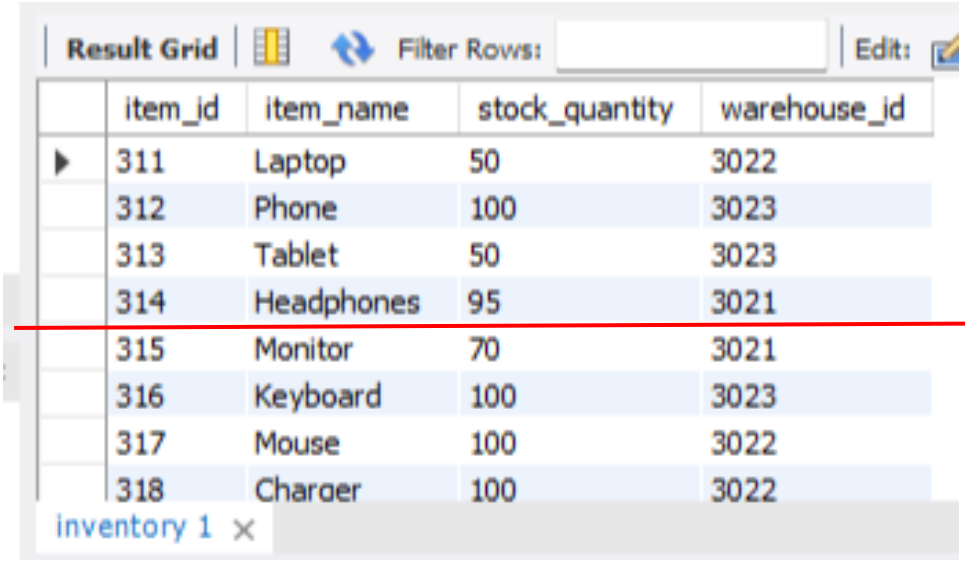
```sql
INSERT INTO Deliveries (delivery_id, vehicle_id, driver_id, warehouse_id, delivery_date, status) VALUES
(411, 101, 1, 3021, '2024-10-14', 'in-progress'),
(412, 102, 2, 3023, '2024-10-15', 'in-progress'),
(413, 103, 3, 3023, '2024-10-16', 'pending'),
(414, 104, 4, 3021, '2024-10-17', 'in-progress'),
(415, 105, 5, 3022, '2024-10-18', 'pending');
```

```sql
INSERT INTO Delivery_Items (delivery_id, item_id, quantity) VALUES
(411, 312, 12),
(411, 315, 20),
(411, 316, 15),
(412, 311, 9),
(412, 312, 12),
(412, 313, 7),
(413, 317, 10),
(413, 318, 15),
(413, 319, 12),
(414, 314, 5),
(414, 320, 7),
(414, 312, 15),
(415, 313, 6),
(415, 311, 19),
(415, 312, 19);
```

3. Basic Data Manipulation

1.Update stock quantities as items are assigned for delivery. For each delivery, reduce the stock in the warehouse for the respective items being delivered

```
UPDATE Inventory
SET stock_quantity = stock_quantity - 5
WHERE item_id = 314 AND warehouse_id = 3021;
```



Result Grid | Filter Rows: | Edit:

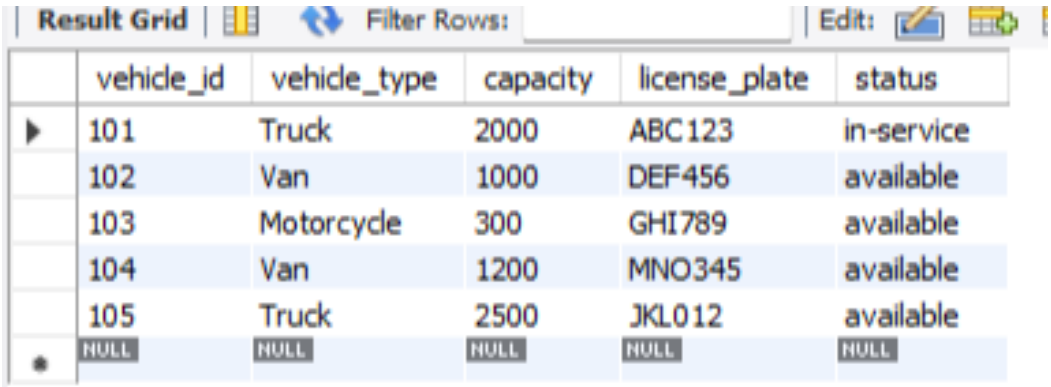
item_id	item_name	stock_quantity	warehouse_id
311	Laptop	50	3022
312	Phone	100	3023
313	Tablet	50	3023
314	Headphones	95	3021
315	Monitor	70	3021
316	Keyboard	100	3023
317	Mouse	100	3022
318	Charger	100	3022

inventory 1

2.Assign deliveries to drivers and update their vehicle status (in-service while delivering, available after the delivery)

Update Vehicle Status: After assigning a delivery, update the vehicle status to "in-service":

```
UPDATE Vehicles
SET status = 'in-service'
WHERE vehicle_id = 101;
```



Result Grid | Filter Rows: | Edit:

vehicle_id	vehicle_type	capacity	license_plate	status
101	Truck	2000	ABC123	in-service
102	Van	1000	DEF456	available
103	Motorcycle	300	GHI789	available
104	Van	1200	MNO345	available
105	Truck	2500	JKL012	available
NULL	NULL	NULL	NULL	NULL

Once the delivery is complete, setting the vehicle status back to "available":

```
UPDATE Vehicles
SET status = 'available'
WHERE vehicle_id = 101;
```

vehicle_id	vehicle_type	capacity	license_plate	status
101	Truck	2000	ABC123	available
102	Van	1000	DEF456	available
103	Motorcycle	300	GHI789	available
104	Van	1200	MNO345	available
105	Truck	2500	JKL012	available
NULL	NULL	NULL	NULL	NULL

<u>Write queries to:</u>

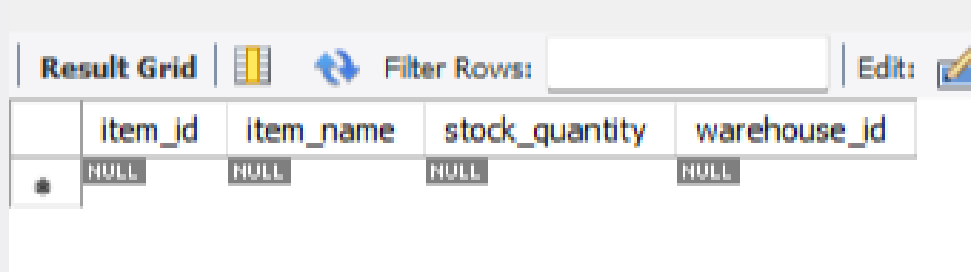
1.List all vehicles currently in service.

```
SELECT * FROM Vehicles
WHERE status = 'in-service';
```

vehicle_id	vehicle_type	capacity	license_plate	status
102	Van	1000	DEF456	in-service
103	Motorcycle	300	GHI789	in-service
NULL	NULL	NULL	NULL	NULL

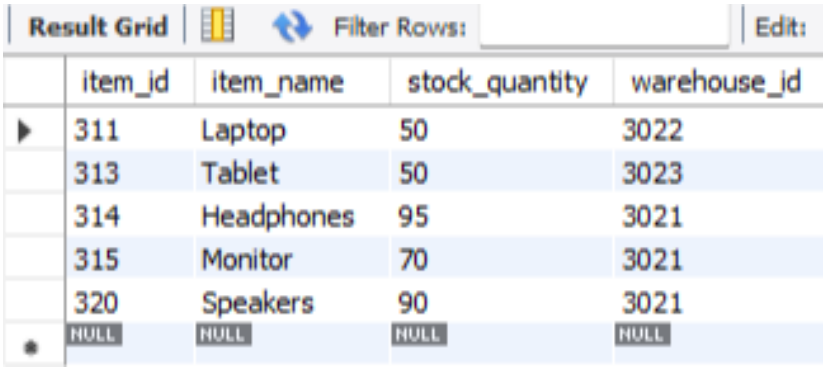
2.Get all items that are low in stock (stock quantity < 10).

```
SELECT * FROM inventory
WHERE stock_quantity < 10;
```



SELECT * FROM inventory

WHERE stock_quantity < 100;



3.List all pending deliveries for a specific day.

SELECT * FROM deliveries

WHERE delivery_date = '2024-10-16' AND status = 'pending';

4. Constraints and Relationships

Add appropriate constraints:

1.Primary keys for each table.

All the tables already have their PRIMARY KEY constraints defined.

2.Foreign keys between related tables (Deliveries and Vehicles, Deliveries and Drivers, Inventory and Warehouses, etc.).

Add foreign key constraints to the Deliveries and Delivery_Items tables to ensure proper relationships between tables.

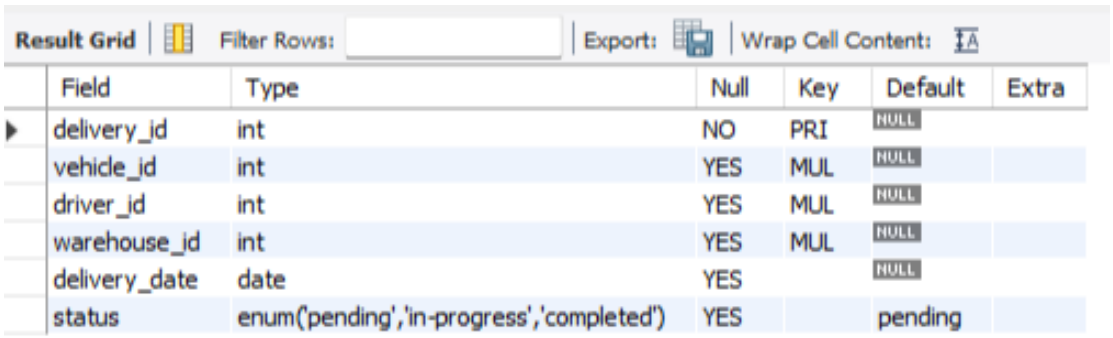
- vehicle_id in Deliveries will reference Vehicles(vehicle_id).
- driver_id in Deliveries will reference Drivers(driver_id).
- warehouse_id in Deliveries will reference Warehouses(warehouse_id).
- delivery_id in Delivery_Items will reference Deliveries(delivery_id).
- item_id in Delivery_Items will reference Inventory(item_id).

ALTER TABLE Deliveries

ADD CONSTRAINT fk_vehicle FOREIGN KEY (vehicle_id) REFERENCES Vehicles(vehicle_id),

ADD CONSTRAINT fk_driver FOREIGN KEY (driver_id) REFERENCES Drivers(driver_id),

ADD CONSTRAINT fk_warehouse FOREIGN KEY (warehouse_id) REFERENCES Warehouses(warehouse_id);

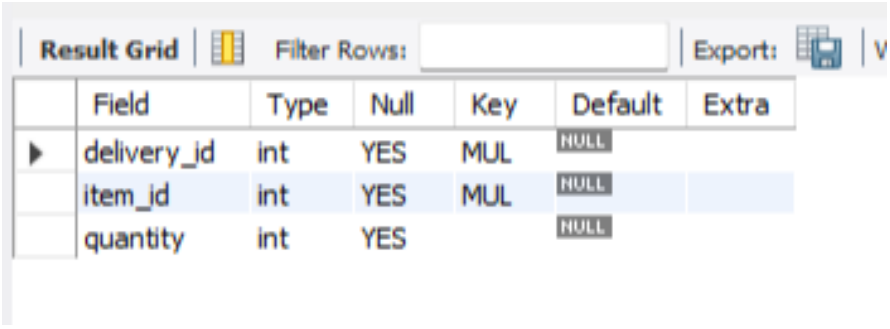


Field	Type	Null	Key	Default	Extra
delivery_id	int	NO	PRI	NULL	
vehicle_id	int	YES	MUL	NULL	
driver_id	int	YES	MUL	NULL	
warehouse_id	int	YES	MUL	NULL	
delivery_date	date	YES		NULL	
status	enum('pending','in-progress','completed')	YES		pending	

ALTER TABLE Delivery_Items

ADD CONSTRAINT fk_delivery FOREIGN KEY (delivery_id) REFERENCES Deliveries(delivery_id),

ADD CONSTRAINT fk_item FOREIGN KEY (item_id) REFERENCES Inventory(item_id);



Field	Type	Null	Key	Default	Extra
delivery_id	int	YES	MUL	NULL	
item_id	int	YES	MUL	NULL	
quantity	int	YES		NULL	

3.Unique constraint on license plates in the Vehicles table.

The unique constraint on the license_plate in the Vehicles table is already defined, ensuring no two vehicles can have the same license plate.

Day2: Advanced Queries, Transactions, and Optimization

<u>Write the following queries:</u>

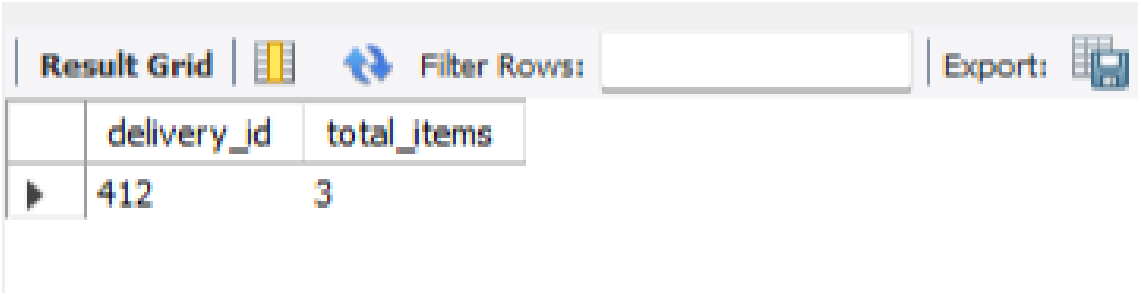
1.List all completed deliveries, including the total number of items delivered.

SELECT d.delivery_id, COUNT(di.item_id) AS total_items FROM Deliveries d

JOIN Delivery_Items di ON d.delivery_id = di.delivery_id

WHERE d.status = 'completed'

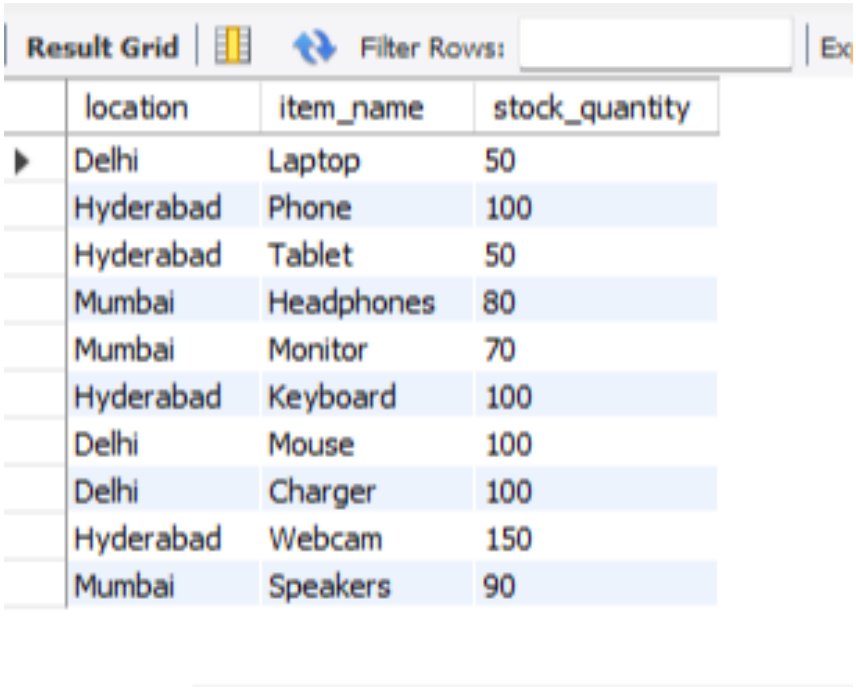
GROUP BY d.delivery_id;



delivery_id	total_items
412	3

2.Get a report showing the current stock levels in each warehouse.

```
SELECT w.location, i.item_name, i.stock_quantity FROM Warehouses w
JOIN Inventory i ON w.warehouse_id = i.warehouse_id;
```



location	item_name	stock_quantity
Delhi	Laptop	50
Hyderabad	Phone	100
Hyderabad	Tablet	50
Mumbai	Headphones	80
Mumbai	Monitor	70
Hyderabad	Keyboard	100
Delhi	Mouse	100
Delhi	Charger	100
Hyderabad	Webcam	150
Mumbai	Speakers	90

3.Generate a report of all drivers and the number of deliveries they've completed.

```
SELECT d.driver_id, d.name, COUNT(dl.delivery_id) AS completed_deliveries
FROM Drivers d
LEFT JOIN Deliveries dl  ON d.driver_id = dl.driver_id
AND dl.status = 'completed'
GROUP BY d.driver_id, d.name
```

ORDER BY d.driver_id;

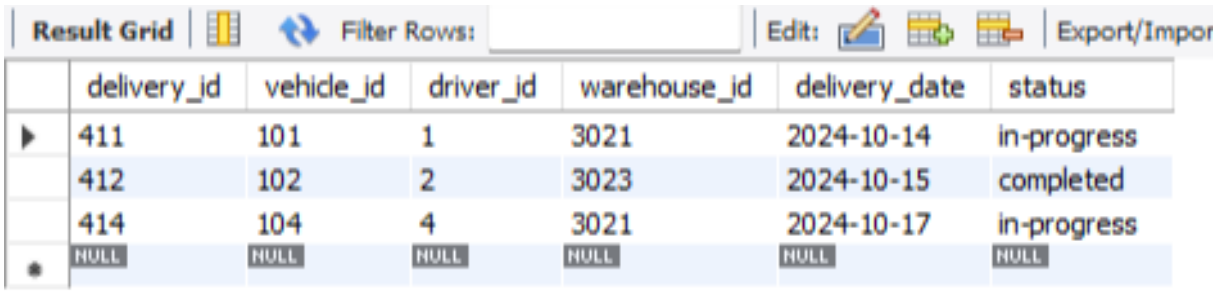
driver_id	name	completed_deliveries
1	John Doe	0
2	Jane Smith	1
3	Tom Brown	0
4	Lucy Green	0
5	Emma White	0

4.Show all deliveries that were in progress or completed within a given time range.

SELECT * FROM Deliveries

WHERE delivery_date BETWEEN '2024-10-01' AND '2024-10-31'

AND (status = 'in-progress' OR status = 'completed');



delivery_id	vehicle_id	driver_id	warehouse_id	delivery_date	status
411	101	1	3021	2024-10-14	in-progress
412	102	2	3023	2024-10-15	completed
414	104	4	3021	2024-10-17	in-progress
NULL	NULL	NULL	NULL	NULL	NULL

6.Identify any deliveries where the total item quantity exceeds the vehicle's capacity (requires a JOIN between Deliveries, Vehicles, and Delivery_Items).

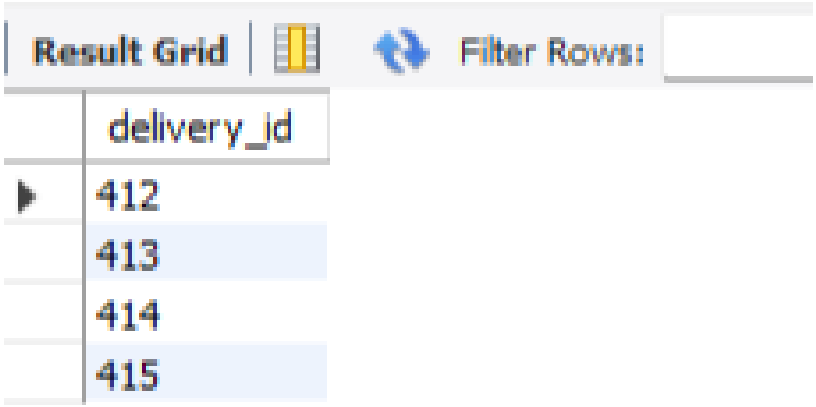
SELECT d.delivery_id FROM Deliveries d

JOIN Delivery_Items di ON d.delivery_id = di.delivery_id

JOIN Vehicles v ON d.vehicle_id = v.vehicle_id

GROUP BY d.delivery_id, v.capacity

HAVING SUM(di.quantity) > v.capacity;



2. Transactions and Error Handling

Implement a transaction to simulate a delivery process:

-- Begin the transaction

START TRANSACTION;

-- 1: Check if there is sufficient stock for each item in the delivery

-- This query will fail if any stock quantity is insufficient, causing the transaction to roll back

UPDATE Inventory i

JOIN Delivery_Items di ON i.item_id = di.item_id

JOIN Deliveries d ON d.delivery_id = di.delivery_id

SET i.stock_quantity = i.stock_quantity - di.quantity

WHERE d.delivery_id = 411 -- specify the delivery_id

AND i.stock_quantity >= di.quantity;

-- 2: Update delivery status to 'in-progress'

-- If the above query doesn't fail, update the delivery status

```
UPDATE Deliveries

SET status = 'in-progress'

WHERE delivery_id = 411;


--  3: Commit the transaction if everything is successful

COMMIT;
```

9	17:55:41	COMMIT

1. UPDATE statement diminishes the stock quantity of all the items dilivered. In case of any item being out of stock (i.e. i.stock_quantity < di.quantity) then query will not update, and transaction will be rolled back.

2. Is the change of the delivery state to ‘in-progress’ after stock is deducted successfully.

However, if any of the stock updates fail due to an insufficient quantity, no part of the transaction will be committed.

3. Stored Procedures

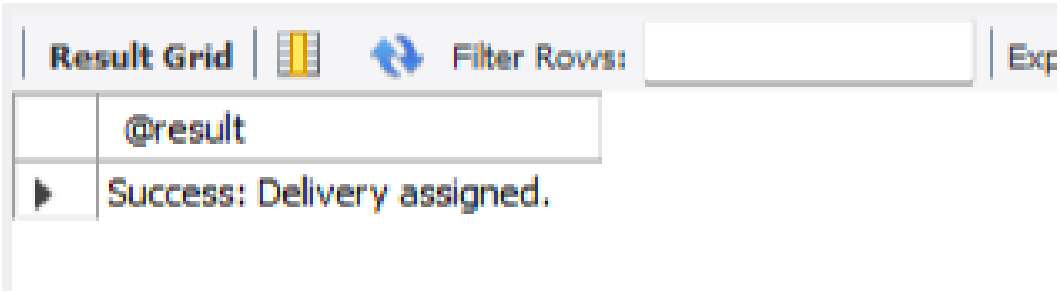
Created a stored procedure to assign deliveries to drivers and vehicles:

Inputs are : delivery_id, vehicle_id, driver_id.

```
DELIMITER $$


CREATE PROCEDURE AssignDelivery (

   IN p_delivery_id INT,

   IN p_vehicle_id INT,
```

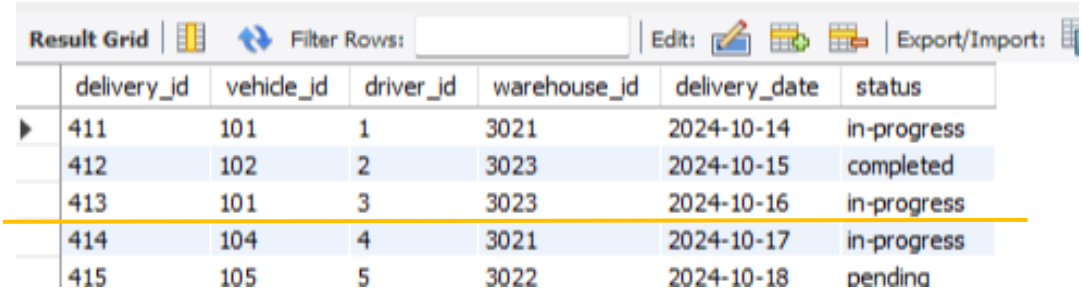
```
    IN p_driver_id INT,
    OUT p_status VARCHAR(50)
)
BEGIN
    DECLARE v_status ENUM('available', 'in-service');

    -- Check if the vehicle is available
    SELECT status INTO v_status FROM Vehicles WHERE vehicle_id = p_vehicle_id;

    IF v_status = 'available' THEN
        -- Assign vehicle and driver to the delivery
        UPDATE Deliveries
        SET vehicle_id = p_vehicle_id,
            driver_id = p_driver_id,
            status = 'in-progress'
        WHERE delivery_id = p_delivery_id;

        -- Update vehicle status to 'in-service'
        UPDATE Vehicles
        SET status = 'in-service'
        WHERE vehicle_id = p_vehicle_id;

        SET p_status = 'Success: Delivery assigned.';
    ELSE
        SET p_status = 'Failure: Vehicle is not available.';
    END IF;
END$$
DELIMITER ;

CALL AssignDelivery(413, 101, 3, @result);
SELECT @result;
```

Output:



Now in deliveries table status is changed from pending to in-process and also In vehicle table from available to in-service.

Deliveries Table



delivery_id	vehicle_id	driver_id	warehouse_id	delivery_date	status
411	101	1	3021	2024-10-14	in-progress
412	102	2	3023	2024-10-15	completed
413	101	3	3023	2024-10-16	in-progress
414	104	4	3021	2024-10-17	in-progress
415	105	5	3022	2024-10-18	pending

Vehicle Table

vehicle_id	vehicle_type	capacity	license_plate	status
101	Truck	50	ABC123	in-service
102	Van	20	DEF456	in-service
103	Motorcycle	1	GHI789	in-service
104	Van	15	MNO345	available
105	Truck	20	JKL012	available
NULL	NULL	NULL	NULL	NULL

This procedure checks if the vehicle is available. If it is, the delivery is assigned to the driver and vehicle, and the vehicle status is updated. If not, an error message is returned.

4. Performance Optimization:

1. Index on delivery date for filtering by date ranges

```
CREATE INDEX idx_delivery_date ON Deliveries(delivery_date);
```

Example:

```
4 •   EXPLAIN SELECT * FROM Deliveries
5     WHERE delivery_date = '2024-10-15' AND status = 'completed';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	Deliveries	NULL	ref	idx_delivery_date	idx_delivery_date	4	const	1	33.33	Using where

This index helps speed up queries that filter or sort by the delivery_date column in the Deliveries table.

2.Indexes for joins between Deliveries and Delivery_Items.

```
CREATE INDEX idx_delivery_id ON Delivery_Items(delivery_id);
CREATE INDEX idx_vehicle_id ON Deliveries(vehicle_id);
```

Example:

```
9
10 •  EXPLAIN SELECT d.delivery_id, COUNT(di.item_id) AS total_items FROM Deliveries d
11    JOIN Delivery_Items di ON d.delivery_id = di.delivery_id
12    WHERE d.status = 'completed'
13    GROUP BY d.delivery_id;
14
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered
1	SIMPLE	d	NULL	index	PRIMARY,fk_driver,fk_warehouse,idx_delivery...	PRIMARY	4	NULL	5	33.33
1	SIMPLE	di	NULL	ref	idx_delivery_id	idx_delivery_id	5	inventory_and_fleet.d.delivery_id	3	100.00

This index will optimize queries that involve the delivery_id column, which is likely used to associate items with their corresponding deliveries.

This index enhances the performance of queries that filter or join on the vehicle_id column in the Deliveries table.